UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/683,929	10/09/2003	John W. Rapp	1934-13-3	2222

7590 12/20/2007
Bryan A. Santarelli
GRAYBEAL JACKSON HALEY LLP
Suite 350
155 - 108th Avenue NE
Bellevue, WA 98004-5901

EXAMINER	
HUISMAN, DAVID J	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
12/20/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

mN

Office Action Summary	**Application No.** 10/683,929	**Applicant(s)** RAPP ET AL.
	Examiner David J. Huisman	**Art Unit** 2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *22 October 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-16,41-50 and 66-85* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-16,41-50 and 66-85* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *11 March 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-16, 41-50, and 66-85 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received subsequent to filing and placed of record in the file: RCE, Amendment, and Extension of Time, as received on 10/22/2007.

Specification

3. The amended title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

Claim Objections

4. Claim 72 is objected to because of the following informalities: In the 3rd to last line, please insert --the-- before "destination". Appropriate correction is required.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1, 4-6, 41-42, 66, and 80 are rejected under 35 U.S.C. 102(b) as being anticipated by Morikawa.

7. Referring to claim 1 (under a first interpretation), Morikawa has taught a pipeline accelerator, comprising:

a) a memory. See Fig.10, and note that input buffer 330/331 and output buffer 332/333 make up a memory (an I/O buffer memory).

b) a hardwired-pipeline circuit (see Fig.10; the portions of coprocessor 302 excluding the buffers; also see the bottom portion of Fig.6 and note that the coprocessor is pipelined) coupled to the memory, including at least one processing pipeline, and operable to:

b1) receive, via a single bus, a message that includes data and that includes a header having information indicating a destination of the data. See Fig.10, and claims 1-2 of Morikawa. The coprocessor receives an instruction and data via a single bus comprising sub-buses 124 and 126. The instruction and data, collectively, are a "message". The header is the portion of the message that specifies the destination of the data (which buffer it is stored in upon arrival in the coprocessor). Note that the message has this information because it is the message which causes the control circuit to operate, and the control circuit operates the coprocessor.

b2) extract the data from the message. The message is simply a number of bit inputs into the coprocessor. Among all of the bits that are inputted, the data consumes just a portion of those bits, and is therefore extracted from among all the bits.

b3) load the extracted data into the memory. Fig.10 shows that the register data coming from the main processor on bus 124 and latch 118 is ultimately put into the memory portion 330/331 by the driver 351/352 of the pipeline circuit.

b3) retrieve the extracted data from the memory. Fig.10 shows that the data is retrieved from the memory portion 330/331 by the coprocessor processing unit 211.

b4) process the retrieved data with a pipeline corresponding to the destination. Once the data has been retrieved from the buffer specified by the destination data, it is processed by the processing pipeline. See column 18, lines 22-27.

b5) provide the processed data to an external source. See column 18, lines 28-32, and Fig.10. Note that after processing, the data is sent to the processor 301, which is external to the coprocessor (hardwired-pipeline circuit).

8. Referring to claim 4, Morikawa has taught a pipeline accelerator as described in claim 1. Morikawa has further taught that the pipeline circuit is operable to provide the processed data to the external source by:

a) loading the processed data into the memory. See Fig.10, and note that after the data is processed, it is stored in memory portion 332/333 through driver 355/356.

b) retrieving the processed data from the memory and providing the retrieved processed data to the external source. Again, see column 18, lines 28-32, and Fig.10. The data is retrieved from the memory portion 332/333 and sent to the processor via bus 125.

9. Referring to claim 5, Morikawa has taught a pipeline accelerator as described in claim 1. Morikawa has further taught that:

a) the external source comprises a processor. See Fig.5, component 301.

b) the pipeline circuit is operable to receive the data from the processor. See Fig.10. The data that is sent to the coprocessor (pipeline circuit) is from the processor.

10. Referring to claim 6, Morikawa has taught a computing machine comprising:

a) a processor operable to broadcast a message that includes data and that includes a header having information indicating a destination of the data. See Fig.10, component 301. The coprocessor receives an instruction via bus 126 and data via bus 124. The instruction and data, collectively, are a "message". The header is the portion of the message that specifies the destination of the data (which buffer it is stored in upon arrival in the coprocessor). Note that the message has this information because it is the message which causes the control circuit to operate, and the control circuit operates the coprocessor.

b) a pipeline accelerator (Fig.10, component 302) coupled to the processor and comprising:

> b1) a memory. See Fig.10, and note that input buffer 330/331 and output buffer 332/333 make up a memory (an I/O buffer memory).
>
> b2) a hardwired-pipeline circuit (see Fig.10; the portions of component 302 excluding the buffers; also see the bottom portion of Fig.6 and note that the coprocessor is pipelined) coupled to the memory, including at least one processing pipeline, and operable to:
>
> - receive, via a single data stream, the message from the processor. See Fig.10 and claims 1-2 of Morikawa. The coprocessor receives the "message" from the processor from a single data stream (over a single bus comprising sub-buses 124 and 126).
> - extract the data from the message. The message is simply a number of bit inputs into the coprocessor. Among all of the bits that are inputted, the data

consumes just a portion of those bits, and is therefore extracted from among all the bits.

- load the extracted data into the memory. Fig.10 shows that the register data coming from the main processor on bus 124 and latch 118 is ultimately put into the memory portion 330/331 by the driver 351/352 of the pipeline circuit.
- retrieve the extracted data from the memory. Fig.10 shows that the data is retrieved from the memory portion 330/331 by the coprocessor processing unit 211.
- process the retrieved data with a pipeline corresponding to the destination. Once the data has been retrieved from the buffer specified by the destination data, it is processed by the processing pipeline. See column 18, lines 22-27.
- provide the processed data to an external source. See column 18, lines 28-32, and Fig.10. Note that after processing, the data is sent to the processor 301, which is external to the coprocessor (hardwired-pipeline circuit).

11. Referring to claim 41, the method of claim 41 is performed by the circuit of claim 1. Consequently, claim 41 is rejected for the same reasons set forth in the rejection of claim 1. Also, the message has information indicating a size of the message. See Fig.2 and note the coprocessor instructions which are part of the message. Each includes an opcode which specifies the size of the immediate data (imm8, imm16, and imm32) of the message.

12. Referring to claim 42, Morikawa has taught a method as described in claim 41. Furthermore, the method of claim 42 is performed by the circuit of claim 4. Consequently, claim 42 is rejected for the same reasons set forth in the rejection of claim 4.

13. Referring to claim 66, Morikawa has taught a pipeline accelerator as described in claim 1. Morikawa has further taught that the hardwired-pipeline circuit is further operable to:

a) extract from the header the information indicating the destination of the data. Note that instructions include destinations so that data may be stored.

b) generate from the extracted information an identifier that identifies the pipeline corresponding to the destination. The opcode portion of the instruction specifies which pipeline unit will be used in the execution. For instance, a floating-point opcode will denote the use of a floating-point pipeline.

c) store the identifier in association with the data. The identifier is inherently stored in the instruction register (IR), a cache, main memory, etc.

d) provide the retrieved data to the pipeline in response to the stored identifier. The opcode specifying a particular pipeline will deal with the retrieved data.

14. Referring to claim 80, Morikawa has taught a method as described in claim 41. Furthermore, the method of claim 80 is performed by the circuit of claim 66. Consequently, claim 80 is rejected for the same reasons set forth in the rejection of claim 66.

15. Claims 1, 4, 41-42, 66, and 80 are rejected under 35 U.S.C. 102(b) as being anticipated by Hennessy and Patterson, "Computer Architecture - A Quantitative Approach, 2nd Edition," 1996 (herein referred to as Hennessy).

16. Referring to claim 1, Hennessy has taught a pipeline accelerator, comprising:

a) a memory. See page 134, Fig.3.4, and note the register file (labeled "Registers").

b) a hardwired-pipeline circuit (the remaining parts of Fig.3.4 excluding the instruction memory and the data memory) coupled to the memory, including at least one processing pipeline, and operable to:

b1) receive, via a single bus, a message that includes data and that includes a header having information indicating a destination of the data. See page 155, Fig.3.15, and note that when the pipeline circuit receives the message "LW R1, B" instruction, the header (opcode) specifies that the destination is a register (in this case R1). The message also includes data (the value B).

b2) extract the data from the message. The value B is extracted from the message so that is can be stored.

b3) load the extracted data into the memory. Again, see page 155, Fig.3.15, and note that the "LW R1, B" instruction loads data B into the register file (location R1).

b3) retrieve the extracted data from the memory. See page 155, Fig.3.15, and note that when the pipeline executes the "ADD R3, R1, R2" instruction, the data value B, which was previously stored in register R1, is now retrieved from R1 so that it may be processed by the ADD instruction.

b4) process the retrieved data with a pipeline corresponding to the destination. The ADD instruction processes the data value B by adding a value to it. The value added to it is the value stored in register R2 (data value C, which was loaded into R2 by the second LW instruction in Fig.3.15).

b5) provide the processed data to an external source. See Fig.3.15 on page 155 and note that after the ADD instruction, the processed data is stored to address A of data memory, which is shown in Fig.3.4 on page 134.

17. Referring to claim 4, Hennessy has taught a pipeline accelerator as described in claim 1. Hennessy has further taught that the pipeline circuit is operable to provide the processed data to the external source by:

a) loading the processed data into the memory. See page 155, Fig.3.15 and note that the ADD instruction, after processing the data B (in R1) will load the result into the register file memory at location R3.

b) retrieving the processed data from the memory and providing the retrieved processed data to the external source. See page 155, Fig.3.15, and note that the SW instruction will retrieve the data from register R3 and provide it to the data memory at address A.

18. Referring to claim 41, the method of claim 41 is performed by the circuit of claim 1. Consequently, claim 41 is rejected for the same reasons set forth in the rejection of claim 1. Also, the message has information indicating a size of the message. The "LW" indicates that a word B is to be loaded into R1. Specifically, B is an immediate value that is equivalent in size to a word. The size of B, which is indicated, is a size of the message.

19. Referring to claim 42, Hennessy has taught a method as described in claim 41. Furthermore, the method of claim 42 is performed by the circuit of claim 4. Consequently, claim 42 is rejected for the same reasons set forth in the rejection of claim 4.

20. Referring to claim 66, Hennessy has taught a pipeline accelerator as described in claim 1. Hennessy has further taught that the hardwired-pipeline circuit is further operable to:

a) extract from the header the information indicating the destination of the data. See page 155 and note that the header in the LW R1, B instruction could comprise "LW R1" which includes the destination R1.

b) generate from the extracted information an identifier that identifies the pipeline corresponding to the destination. The LW portion of the instruction is the opcode which specifies that a load pipeline will perform the load.

c) store the identifier in association with the data. The identifier is inherently stored in the instruction register (IR), a cache, main memory, etc.

d) provide the retrieved data to the pipeline in response to the stored identifier. The opcode specifying the load pipeline will deal with the load data.

21. Referring to claim 80, Hennessy has taught a method as described in claim 41. Furthermore, the method of claim 80 is performed by the circuit of claim 66. Consequently, claim 80 is rejected for the same reasons set forth in the rejection of claim 66.

Claim Rejections - 35 USC § 103

22. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

23. Claim 2 is rejected under 35 U.S.C. 103(a) as being unpatentable over Morikawa in view of Tubbs et al., U.S. Patent No. 5,752,071 (herein referred to as Tubbs).

24. Referring to claim 2, Morikawa has taught a pipeline accelerator as described in claim 1 (under the second interpretation). Morikawa has taught that the coprocessor and memory (register file in the processor) are disposed on a single chip (Fig.11), and has not taught that the memory is disposed on a first integrated circuit and the pipeline circuit is disposed on a second integrated circuit. However, Tubbs has taught that a coprocessor and processor (which includes the claimed memory) are disposed on separate integrated circuits. See column 8, lines 51-52. A person of ordinary skill in the art would have recognized that by disposing the two components on different chips, increased flexibility is realized. For example, if one chip goes bad, it may be replaced without having to replace the other. Or, if an upgrade to one of the components is desired, the chip may be swapped out for an upgraded chip without affecting the other. In addition, as shown in <u>Nerwin v. Erlichman</u> *168 USPQ 177 (1969)*, to make separable (i.e., to turn a single chip into two chips) is generally not given patentable weight or would have been an obvious improvement. As a result, in order to increase flexibility, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morikawa such that the memory and pipeline circuit are disposed on separate integrated circuits.

25. Claims 3, 12, and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Morikawa in view of FOLDOC.

26. Referring to claim 3, Morikawa has taught a pipeline accelerator as described in claim 1 (under both the first and second interpretations). Morikawa has not taught that the pipeline circuit is disposed on a field-programmable gate array (FPGA). However, FOLDOC has taught that FPGAs are devices which are programmed after manufacture time. See the 1st paragraph of

FOLDOC. This is clearly an advantage as designers are able to reprogram/reconfigure the device after it has been made. That is, if any new functionality is desired, a new chip would not have to be manufactured. Instead, the associated functionality would simply be programmed into the FPGA. Consequently, in order to make the pipeline circuit of Morikawa reconfigurable and, consequently, more flexible, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morikawa such that the pipeline is disposed on an FPGA as taught by FOLDOC.

27. Referring to claim 12, Morikawa has taught a pipeline accelerator as described in claim 9.

a) Morikawa has not taught that the first and second memories are respectively disposed on first and second integrated circuits. However, as shown in Nerwin v. Erlichman *168 USPQ 177 (1969)*, to make separable is generally not given patentable weight or would have been an obvious improvement. In this case, it would have been obvious to have the buffers of the coprocessor of Fig.10 to be on separate chips. For instance, everything prior to the processing unit 211 would be on one chip, while the rest would be on another. A person of ordinary skill in the art would have recognized that separating the components into multiple chips allows for more flexibility and cost savings. For example, if the coprocessor were divided into multiple chips and one chip is malfunctioning or is defective, just that chip would need to be replaced while the other functioning chip would be left in tact. This results in not having to spend money replacing the functioning chip. Another advantage would be to gain the ability to upgrade a portion of the coprocessor. As a result, in order to increase flexibility, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morikawa to include the first and second memories on separate integrated circuits.

b) Morikawa has not taught that the pipeline circuit is disposed on a field-programmable gate array (FPGA). However, FOLDOC has taught that FPGAs are devices which are programmed after manufacture time. See the 1st paragraph of FOLDOC. This is clearly an advantage as designers are able to reprogram/reconfigure the device after it has been made. That is, if any new functionality is desired, a new chip would not have to be manufactured. Instead, the associated functionality would simply be programmed into the FPGA. Consequently, in order to make the pipeline circuit of Morikawa reconfigurable and, consequently, more flexible, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morikawa such that the pipeline is disposed on an FPGA, as taught by FOLDOC.

28. Referring to claim 15, Morikawa has taught a pipeline accelerator as described in claim 9.

a) Morikawa has further taught that each of the input-data handler, hardwired pipeline, pipeline interface, and output-data handler has a respective operating configuration. That is, each component in Morikawa operates in a specific fashion and consequently, each component inherently has a respective operating configuration.

b) Morikawa has not taught a configuration manager coupled to and operable to set the operating configurations of the input-data handler, hardwired pipeline, pipeline interface, and output-data handler. However, FOLDOC has taught that FPGAs are devices which are programmed after manufacture time. See the 1st paragraph of FOLDOC. This is clearly an advantage as designers are able to reprogram/reconfigure the device after it has been made. That is, if any new functionality is desired, a new chip would not have to be manufactured. Instead, the associated functionality would simply be programmed into the FPGA. Consequently, in order to make the pipeline circuit of Morikawa reconfigurable and, consequently, more flexible, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morikawa such that the system is disposed on an FPGA. Furthermore, an FPGA would be coupled to a configuration manager so that the FPGA may be programmed to include the desired functionality.

29. Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Hennessy in view of FOLDOC.

30. Referring to claim 3, Hennessy has taught a pipeline accelerator as described in claim 1. Hennessy has not taught that the pipeline circuit is disposed on a field-programmable gate array (FPGA). However, FOLDOC has taught that FPGAs are devices which are programmed after manufacture time. See the 1st paragraph of FOLDOC. This is clearly an advantage as designers are able to reprogram/reconfigure the device after it has been made. That is, if any new functionality is desired, a new chip would not have to be manufactured. The associated functionality would simply be programmed into the FPGA. Consequently, in order to make the system of Hennessy reconfigurable and, consequently, more flexible, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hennessy such that the pipeline of page 134 is disposed on an FPGA as taught by FOLDOC.

31. Claims 7-8, 43, 69-72, and 83-85 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grondalski, U.S. Patent No. 4,985,832.

32. Referring to claim 7, Grondalski has taught an accelerator, comprising:

a) a memory. See Fig.3, components 22A, for instance.

b) a hard-wired circuit coupled to the memory (see Fig.3, and note the processor coupled to memory 22A) and operable to,

b1) receive data. Every processor must inherently received data in order to process it.

b2) process the received data. Every processor also inherently processes data received; otherwise, no useful work is performed.

b3) load the processed data into the memory. See column 1, lines 17-31, and note the general operation of a processor and memory. When a processor finishes processing data, it is written to some memory.

b4) retrieve the processed data from the memory. See the abstract, column 1, lines 26-31, column 8, lines 44-48, column 8, line 59, to column 9, line 17, and Fig.2. Note that when data is to be transferred to another processor, it will be retrieved from memory, where it was stored after processing. Note that it must have been stored because transfer of the data may not happen immediately. It may only occur when a particular flag is set.

b5) generate a message header that includes first information including a destination of the processed data, forming a message from the header and the processed data and the header, and providing the message to an external source via a single bus. See Fig.8b and column 25, lines 5-9.

c) Grondalski has not explicitly taught that his processors of Fig.3 are pipelined. However, Grondalski has taught that processors are known to incorporate pipelining. See column 2, lines 45-47. Pipelining has known advantages such as increasing throughput by overlapping execution of instruction as opposed to serial execution of instructions which is slower.

Consequently, in order to increase throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Grondalski to include pipelined processors.

33. Referring to claim 8, Grondalski has taught a computing machine comprising:

a) a processor operable to run at least one software application. See Fig.3, and note that any of the processors may qualify as the processor, and note that processors inherently run a software application.

b) an accelerator coupled to the processor (Fig.3, the accelerator is any one or more processors that are not the processor above) and comprising:

b1) a memory. See Fig.3, component 22A, for instance.

b2) a hardwired circuit coupled to the memory (see the processor coupled to memory 22A in Fig.3) and operable to:

- receive data from the processor. See the abstract and note that a processor receives data from the processor.
- process the received data. Processors inherently process data, otherwise, no useful work is performed.
- load the processed data into the memory. See column 1, lines 17-31, and note the general operation of a processor and memory. When a processor finishes processing data, it is written to some memory.
- retrieve the processed data from the memory and provide the retrieved processed data to the processor. See the abstract, column 1, lines 26-31, column 8, lines 44-48, column 8, lines 59, to column 9, line 17, and Fig.2. Note that when data is to be transferred to another processor, it will be

retrieved from memory, where it was stored after processing. Note that it must have been stored because transfer of the data may not happen immediately. It may only occur when a particular flag is set.

- generate a message header that includes first information including a destination of the processed data, forming a message from the header and the processed data and the header, and providing the message to an external source via a single bus. See Fig.8b and column 25, lines 5-9.

c) Grondalski has not explicitly taught that the accelerator is pipelined. However, Grondalski has taught that processors are known to incorporate pipelining. See column 2, lines 45-47. Pipelining has known advantages such as increasing throughput by overlapping execution of instruction as opposed to serial execution of instructions which is slower. Consequently, in order to increase throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Grondalski to include pipelined processors.

34. Referring to claim 43, the method of claim 43 is performed by the accelerator of claim 7. Consequently, claim 43 is rejected for the same reasons set forth in the rejection of claim 7. In addition, the providing the message to an external source comprises providing on a single bus. See Fig.2 and note the output bus from the right side of chip 20A.

35. Referring to claim 69, Grondalski, as modified, has taught the pipeline accelerator as described in claim 7. Grondalski has further taught that the hardwired-pipeline circuit is further operable to:

a) store in association with the processed data second information indicating the destination of the processed data. See Fig.8b, and note that the element destination is stored.

b) generate the message header in response to the second information. See Fig.8b and note that in response to the destination, the router switch data is also inserted into the header.

36. Referring to claim 70, Grondalski, as modified, has taught a pipeline accelerator as described in claim 69. Grondalski has further taught that the second information equals the first information. Both the first and second information have been interpreted as being destination processor information. Hence, they are the same.

37. Referring to claim 71, Grondalski has taught:

a) a memory. See Fig.3, components 22A, for instance.

b) a hard-wired circuit coupled to the memory (see Fig.3, and note the processor coupled to memory 22A) and operable to,

b1) receive data. Every processor must inherently received data in order to process it.

b2) process the received data. Every processor also inherently processes data received; otherwise, no useful work is performed.

b3) load the processed data into the memory. See column 1, lines 17-31, and note the general operation of a processor and memory. When a processor finishes processing data, it is written to some memory.

b4) retrieve the processed data from the memory. See the abstract, column 1, lines 26-31, column 8, lines 44-48, column 8, lines 59, to column 9, line 17, and Fig.2. Note that when data is to be transferred to another processor, it will be retrieved from memory, where it was stored after processing. Note that it must have been stored because transfer of the data may not happen immediately. It may only occur when a particular flag is set.

b5) generate a message header that includes first information including a destination of the processed data, forming a message from the header and the processed data and the header, and providing the message to an external source. See Fig.8b and column 25, lines 5-9.

b6) Grondalski has not explicitly taught storing a pointer to the processed data. However, the examiner asserts that a memory address/pointer is known to be stored somewhere in the system so that memory may be accessed at the location pointed to. That is, it is known at the very least to latch a memory address when using it to access data. As is known, latching allows for synchronous operation which is reliable and deterministic operation. Consequently, in order to achieve synchronous operation, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Grondalski such that a pointer to the extracted data is stored.

b7) Grondalski, as modified, has further taught storing in association with the pointer second information indicating the destination of the processed data. See the processor ID field of Fig.8B.

b8) Grondalski, as modified, has taught retrieving the processed data in response to the pointer. Clearly, by indexing the memory with the pointer, the data will be located and retrieved for processing.

b9) Grondalksi, as modified, has further taught generating the message header in response to the second information. See Fig.8b and note the router control fields. The router control fields must be set appropriately such that he message reaches its destination set forth in the processor ID field.

c) Grondalski has not explicitly taught that his processors of Fig.3 are pipelined. However, Grondalski has taught that processors are known to incorporate pipelining. See column 2, lines 45-47. Pipelining has known advantages such as increasing throughput by overlapping execution of instruction as opposed to serial execution of instructions which is slower. Consequently, in order to increase throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Grondalski to include pipelined processors.

38. Referring to claim 72, Grondalski has taught:

a) a memory. See Fig.3, components 22A, for instance.

b) a hard-wired circuit coupled to the memory (see Fig.3, and note the processor coupled to memory 22A) and operable to,

b1) receive data. Every processor must inherently received data in order to process it.

b2) process the received data. Every processor also inherently processes data received; otherwise, no useful work is performed.

b3) load the processed data into the memory. See column 1, lines 17-31, and note the general operation of a processor and memory. When a processor finishes processing data, it is written to some memory.

b4) retrieve the processed data from the memory. See the abstract, column 1, lines 26-31, column 8, lines 44-48, column 8, lines 59, to column 9, line 17, and Fig.2. Note that when data is to be transferred to another processor, it will be retrieved from memory, where it was stored after processing. Note that it must have been stored because transfer of the data may not happen immediately. It may only occur when a particular flag is set.

b5) generate a message header that includes first information indicating a destination of the processed data, generating a message that includes the processed data and the header, and providing the message to an external source. See Fig.8b and column 25, lines 5-9.

b6) Grondalski has not explicitly taught storing a pointer to the processed data in a location associated with destination of the processed data. However, the examiner asserts that a memory address/pointer is known to be stored somewhere in the system so that memory may be accessed at the location pointed to. That is, it is known at the very least to latch a memory address when using it to access data. As is known, latching allows for synchronous operation which is reliable and deterministic operation. Consequently, in order to achieve synchronous operation, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Grondalski such that a pointer to the extracted data is stored.

b7) Grondalski, as modified, has taught retrieving the processed data in response to the pointer. Clearly, by indexing the memory with the pointer, the data will be located and retrieved for processing.

b9) Grondalksi, as modified, has further taught generating the message header in response to the location. The address stored at the location causes the fetching of the data to be inserted in the message.

c) Grondalski has not explicitly taught that his processors of Fig.3 are pipelined. However, Grondalski has taught that processors are known to incorporate pipelining. See column 2, lines 45-47. Pipelining has known advantages such as increasing throughput by overlapping execution of instruction as opposed to serial execution of instructions which is slower.

Consequently, in order to increase throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Grondalski to include pipelined processors.

39. Referring to claim 83, Grondalski, as modified, has taught a method as described in claim 43. Grondalski has further taught:

a) storing in association with the processed data second information indicating the destination of the processed data. See Fig.8b, and note that the element destination is stored.

b) wherein generating the header comprises generating the header in response to the second information. See Fig.8b and note that in response to the destination, the router switch data is also inserted into the header.

40. Referring to claim 84, the method of claim 84 is performed by the accelerator of claim 71. Consequently, claim 84 is rejected for the same reasons set forth in the rejection of claim 71.

41. Referring to claim 85, the method of claim 85 is performed by the accelerator of claim 72. Consequently, claim 85 is rejected for the same reasons set forth in the rejection of claim 72.

42. Claims 9-10, 44-45, 49, and 73 are rejected under 35 U.S.C. 103(a) as being unpatentable over Morikawa.

43. Referring to claim 9, Morikawa has taught a pipeline accelerator comprising:

a) first and second memories. See Fig.10, components 330/331 and 332/333 (input and output buffers).

b) a hardwired-pipeline circuit (see Fig.10; the portions of component 302 excluding the buffers; also see the bottom portion of Fig.6 and note that the coprocessor is pipelined) coupled to the first and second memories and comprising:

b1) an input-data handler operable to receive from an external source a first message that includes raw data and that includes a first header having information indicating a destination of the raw data, to extract the raw data from the message, and to load the raw data into the first memory. See Fig.10, and claims 1-2 of Morikawa. The coprocessor receives an instruction via bus 126 and data via bus 124. The instruction and data, collectively, are a "message". The header is the portion of the message that specifies the destination of the data (which buffer it is stored in upon arrival in the coprocessor). Note that the message has this information because it is the message which causes the control circuit to operate, and the control circuit operates the coprocessor. The message is simply a number of bit inputs into the coprocessor. Among all of the bits that are inputted, the data consumes just a portion of those bits, and is therefore extracted from among all the bits. Furthermore, it should be noted that the portions of the system receiving, controlling receiving, and loading data make up the input data handler.

b2) at least one hardwired pipeline operable to process data. See Fig.10, component 211 and the bottom portion of Fig.6.

b3) a pipeline interface operable to retrieve the raw data from the first memory. See Fig.4, and note that before the data is processed by unit 211, it must be retrieved from the first memory 330/331. The portion of the system retrieving the data would be the pipeline interface.

b4) provide the retrieved raw data to the hardwired pipeline corresponding to the destination. See Fig.4, and note that the data from the first memory 330/331 is provided to the hardwired pipeline 211.

b5) load processed data from the hardwired pipeline into the second memory. See Fig.4, and note that after the data is processed, it is loaded into output buffer 332/333.

b6) an output-data handler operable to retrieve the processed data from the second memory and to provide the processed data to the external source. See column 18, lines 28-32, and Fig.10. The data is retrieved from the memory portion 332/333 and sent to the processor via driver 357/358 and bus 125. The portion of the system controlling the outputting is the output data handler.

b7) Morikawa has not explicitly taught generating a second header having first information indicating a destination of the processed data, generating a second message that includes the processed data and the second header, and providing the second message to the external source. However, Morikawa has taught sending the data from memory 332/333 to the processor (external source). Specifically, the data is written to the register file (the software application context) of the processor. One of ordinary skill in the art would've recognized that if data is to be written to a register file, then a destination register must be specified. In this case, either the processor can specify the destination or the coprocessor can specify the destination. The specifics of such a specification are not disclosed by Morikawa. However, since the coprocessor is performing the operation, the processor can specify the destination as part of a message instead of the processor so that the processor determines the destination only when the coprocessor is finished. This would also allow the processor to save resources for other tasks (not determining the destination). Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morikawa such that the pipeline circuit is operable to generate a message header that includes first information including a destination of the processed data, generate a message that includes the processed data and the header, and provide the message to an external source.

44. Referring to claim 10, Morikawa has taught a pipeline accelerator as described in claim 9. Morikawa has further taught that:

a) the first and second memories each include respective first and second ports. See Fig.10 and note that each of the memories 330/331 and 332/333 include an input and output port.

b) the input-data handler is operable to load the raw data via the first port of the first memory. See Fig.10 and note that data is loaded into first memory 330/331 via its input port.

c) the pipeline interface is operable to retrieve the raw data via the second port of the first memory and to load the processed data via the first port of the second memory. See Fig.10 and note that the data is read from memory 330/331 via its output port, it is processed by component 211, and then the processed data is written to the second memory 332/333 via its input port.

d) the output-data handler is operable to retrieve the processed data via the second port of the second memory. See Fig.10 and note that when the data is to be sent to the processor 301, it is first retrieved from memory 332/333 via its output port.

45. Referring to claim 44, the method of claim 44 is performed by the circuit of claim 9. Consequently, claim 44 is rejected for the same reasons set forth in the rejection of claim 9.

46. Referring to claim 45, Morikawa has taught a method as described in claim 44. Furthermore, the method of claim 45 is performed by the circuit of claim 10. Consequently, claim 45 is rejected for the same reasons set forth in the rejection of claim 10.

47. Referring to claim 49, Morikawa has taught a method as described in claim 44. Furthermore, Morikawa has inherently taught setting parameters for loading and retrieving the raw data, processing the retrieved data, and loading and providing the processed data. That is, if the system is going to load data from the processor, process the data, and then provide data back to the processor, then the system must set which data will be loaded (i.e., which register the data will be coming from), it must set which operation is to be performed on the data, and it must set which register will be receiving the processed data result. Without these parameters, the functions could not be performed.

48. Referring to claim 73, Morikawa has taught a pipeline accelerator as described in claim 9. Furthermore, the input data handler of claim 73 performs in equivalent fashion to the circuit of claim 66. Consequently, claim 73 is rejected for the same reasons set forth in the rejection of claim 66.

49. Claims 11 and 46 are rejected under 35 U.S.C. 103(a) as being unpatentable over Morikawa in view of Fette.

50. Referring to claim 11, Morikawa has taught a pipeline accelerator as described in claim 9. Morikawa has not taught a third memory coupled to the hardwired-pipeline circuit, wherein the hardwired pipeline is operable to generate intermediate data while processing the raw data, and wherein the pipeline interface is operable to load the intermediate data into the third memory and to retrieve the intermediate data from the third memory. However, Fette has taught such a concept. Specifically, Fette has taught that coprocessors contain multiply-accumulate circuitry. See the last sentence of the abstract. A multiply-accumulator is known to include an

accumulation register (third memory). With a multiply-accumulate (MAC) operation, values (raw data) are multiplied to generate a multiplication result. This result is then added to the value already in the accumulation register to generate intermediate data. This intermediate data is then stored in the third memory (back into the accumulator), where it is later retrieved and added to the next multiplication result. A MAC operation is a common operation, and it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morikawa such that the coprocessor performs MAC operations and therefore includes the claimed third memory and its purpose. One would have been motivated to make such a combination to increase the functionality of the coprocessor by giving it MAC functionality.

51. Referring to claim 46, Morikawa has taught a method as described in claim 44. Furthermore, the method of claim 46 is performed by the circuit of claim 11. Consequently, claim 46 is rejected for the same reasons set forth in the rejection of claim 11.

52. Claims 13-14 and 47-48 are rejected under 35 U.S.C. 103(a) as being unpatentable over Morikawa in view of Frey.

53. Referring to claim 13, Morikawa has taught a pipeline accelerator as described in claim 9. Morikawa has not taught an input-data queue coupled to the input-data handler and the pipeline interface, wherein the input-data handler is operable to load into the input-data queue a pointer to a location of the raw data within the first memory and wherein the pipeline interface is operable to retrieve the raw data from the location using the pointer. However, Frey has taught such a concept. See Fig.3, components 17 and 21, and column 11, line 67, to column 12, line 5. Essentially, addresses of operands in a multi-operand buffer are stored in an operand fetch queue

so that when it is time to fetch the operands, they are located appropriately. A person of ordinary skill in the art would have recognized that by implementing a multi-operand buffer in Morikawa (for buffer 330/331), multiple operands would be buffered at a time, which ensures that the coprocessor always has enough work to do. Also, if the coprocessor were to stall in any way, the multi-operand buffer would still be able to buffer operands from the processor, thereby allowing the processor to continue on with other work. With such a system, saving the pointer of an operand allows for the location of the operand in the buffer. Clearly, when an instruction is to operate on an operand, the correct operand should be located, and so the location of the operand must be remembered. Therefore, in order to buffer more operands, ensure work, and avoid stalls, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morikawa to include a multi-operand input buffer and an input-data queue for holding a pointer to an operand (raw data) within the buffer and then using the pointer to locate the desired operand.

54. Referring to claim 14, Morikawa has taught a pipeline accelerator as described in claim 9. Morikawa has not taught an output-data queue coupled to the output-data handler and the pipeline interface wherein the pipeline interface is operable to load into the output-data queue a pointer to a location of the processed data within the second memory and wherein the output-data handler is operable to retrieve the processed data from the location using the pointer. However, Frey has taught keeping a queue of pointers so that data in a multiple-entry buffer would be appropriately located. See Fig.3, components 17 and 21, and column 11, line 67, to column 12, line 5. A person of ordinary skill in the art would have recognized that by implementing a multi-result buffer in Morikawa (for buffer 332/333), multiple results would be

buffered at a time, which ensures that the coprocessor would function smoothly if bus contention were present. For instance, the regular processor may be trying to write to its own register file. If this is the case, the coprocessor cannot write a result to the register file at the same time. Consequently, if only a single entry output buffer exists, then the result must be stored in that buffer until the register file is available. Until it's available, another result cannot be produced as the previous result is already in the buffer. This would result in stalling the coprocessor. With a multiple-entry buffer, the coprocessor would be able to continue executing and storing results until the register file becomes available. And, with such a system, saving the pointer of the result allows for the location of the result in the buffer. Clearly, when a result is to be written to the register file, the correct result should be written, and so the location of the result must be remembered. Therefore, in order to avoid stall potential, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morikawa to include a multiple-entry output buffer and an output-data queue for holding a pointer to a result (processed data) within the buffer and then using the pointer to locate the desired data.

55. Referring to claim 47, Morikawa has taught a method as described in claim 44. Furthermore, the method of claim 47 is performed by the circuit of claim 13. Consequently, claim 47 is rejected for the same reasons set forth in the rejection of claim 13.

56. Referring to claim 48, Morikawa has taught a method as described in claim 44. Furthermore, the method of claim 48 is performed by the circuit of claim 14. Consequently, claim 48 is rejected for the same reasons set forth in the rejection of claim 14.

57. Claims 16 and 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Morikawa.

58. Referring to claim 16, Morikawa has taught a method as described in claim 9.

a) Morikawa has inherently taught that each of the input-data handler, hardwired pipeline, pipeline interface, and output-data handler has a respective operating status. That is, at the very least, each component in the system is either operating or not operating (and these are statuses).

b) Morikawa has not taught an exception manager coupled to and operable to identify an exception in the input-data handler, hardwired pipeline, pipeline interface, or output-data handler in response to the operating statuses. However, Official Notice is taken that checking for errors in a pipeline during processing is well known and accepted in the art. For instance, if a coprocessor is performing a divide operation, which is a well known operation performed by coprocessors, then a division of a number by 0 should raise an exception, as it is an illegal operation. This type of error should be monitored so that the system can take appropriate action to correct the error. As a result, in order to ensure proper execution, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morikawa such that the coprocessor of Morikawa performs divide operations (in order to obtain the ability to perform division) and exceptions are detected in a hardwired pipeline.

59. Referring to claim 50, Morikawa has taught a method as described in claim 44. While Morikawa has not explicitly taught determining whether an error occurs during the loading and retrieving of the raw data, the processing of the retrieved data, and the loading and providing of the processed data, Official Notice is taken that checking for errors during processing is well known and accepted in the art. For instance, if a coprocessor is performing a divide operation,

which is a well known operation performed by coprocessors, then a division of a number by 0 should raise an error as it is an illegal operation. This type of error should be monitored so that the system can take appropriate action to correct the error. As a result, in order to ensure proper execution, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Morikawa such that the coprocessor of Morikawa performs divide operations (in order to obtain the ability to perform division) and errors are detected during the loading and retrieving of the raw data, the processing of the retrieved data, and the loading and providing of the processed data.

60. Claims 1, 67-68, and 81-82 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nakagoshi et al., U.S. Patent No. 5,377,333 (herein referred to as Nakagoshi).

61. Referring to claim 1, Nakagoshi has taught an accelerator, comprising:

a) a memory. See Fig.3, component 203.

b) a hardwired circuit coupled to the memory, including at least one processing unit (see Fig.3, at least components 201), and operable to:

b1) receive, via a single bus, a message that includes data and that includes a header having information indicating a destination of the data. See Fig.5.

b2) extract the data from the message. See column 11, lines 1-14.

b3) load the extracted data into the memory. See column 7, lines 39-53.

b4) retrieve the extracted data from the memory. The data must be retrieved from memory to be processed.

b5) process the retrieved data with a unit corresponding to the destination. See Fig.3 and note the computation units which do processing.

b6) Nakagoshi has not explicitly taught providing the processed data to an external source. However, the examiner asserts that it is well known that processors, when done processing, store data to memory (i.e., an external source). This allows a large amount of processed data to be stored, which is useful if needed at some subsequent point during processing. As a result, in order to store results for future use, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi such that the processed data is provided to an external source.

c) Nakagoshi has not taught a pipelined implementation. However, pipelining is well known and accepted in the art. Pipelining has known advantages such as increasing throughput by overlapping execution of instruction as opposed to serial execution of instructions which is slower. Consequently, in order to increase throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi to be pipelined.

62. Referring to claim 67, Nakagoshi, as modified, has taught the pipeline accelerator of claim 1, wherein the hardwired pipeline circuit is further operable to:

a) extract from the header the information indicating the destination of the data. See Fig.5, at least field 501.

b) generate from the extracted information an identifier that identifies the pipeline corresponding to the destination. Based on the type of processing determined by field 503 (column 7, lines 44-47), the appropriate hardware is determined. For instance, if integer processing is required, as is well known, the identifier will specify integer processing, and the data will be sent to the integer unit. However, if floating-point processing is required, as is well known, then it will be sent to the floating-point unit.

c) Nakagoshi has not explicitly taught storing a pointer to the extracted data. However, the examiner asserts that a memory address/pointer is known to be stored somewhere in the system so that memory may be accessed at the location pointed to. That is, it is known at the very least to latch a memory address when using it to access data. As is known, latching allows for synchronous operation which is reliable and deterministic operation. Consequently, in order to achieve synchronous operation, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi such that a pointer to the extracted data is stored.

d) store the identifier in association with the pointer. See Fig.5, field 503.

e) provide the retrieved data to the circuit in response to the stored pointer and identifier. In order for the data to be sent for processing, it must be retrieved from the memory using the stored pointer. And, based on the type of processing, the appropriate hardware is determined. For instance, if integer processing is required, as is well known, the identifier will specify integer processing, and the data will be sent to the integer unit. However, if floating-point processing is required, as is well known, then it will be sent to the floating-point unit.

63. Referring to claim 68, Nakagoshi has taught an accelerator, comprising:

a) a memory. See Fig.2, component 203.

b) a hardwired circuit coupled to the memory (Fig.1 and Fig.2, at least components 201), and operable to:

b1) receive, via a single bus, a message that includes data and that includes a header having information indicating a destination of the data. See Fig.5.

b2) extract the data from the message. See Fig.5 and note that data is extracted from the message so that it can be operated upon.

b3) load the extracted data into the memory. See column 7, lines 49-52 ·

b3) retrieve the extracted data from the memory. See Fig.1 and column 7, lines 45-47 and note that the processor coupled to the memory fetches the data from memory and processes it as indicated in the message.

b4) process the retrieved data with a pipeline corresponding to the destination. The data that is to be processed as specified does correspond to the destination it is stored in.

b5) extract from the header the information indicating the destination of the data. See Fig.5.

b6) Nakagoshi has not explicitly taught storing a pointer to the extracted data in a location associated with the pipeline corresponding to the destination. However, the examiner asserts that a memory address/pointer is known to be stored somewhere in the system so that memory may be accessed at the location pointed to. That is, it is known at the very least to latch a memory address when using it to access data. As is known, latching allows for synchronous operation which is reliable and deterministic operation. Consequently, in order to achieve synchronous operation, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi such that a pointer to the extracted data is stored.

b7) Nakagoshi, as modified, has further taught providing the retrieved data to the processor in response to the stored pointer. Clearly, by indexing the memory with the pointer, the data will be located and retrieved for processing.

b8) Nakagoshi has not explicitly taught providing the processed data to an external source. However, the examiner asserts that it is well known that processors, when done processing, store data to memory (i.e., an external source). This allows a large amount of processed data to be stored, which is useful if needed at some subsequent point during processing. As a result, in order to store results for future use, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi such that the processed data is provided to an external source.

c) Nakagoshi has not taught a pipelined implementation. However, pipelining is well known and accepted in the art. Pipelining has known advantages such as increasing throughput by overlapping execution of instruction as opposed to serial execution of instructions which is slower. Consequently, in order to increase throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi to be pipelined.

64. Referring to claim 81, the method of claim 81 is performed by the accelerator of claim 1. Consequently, claim 81 is rejected for the same reasons set forth in the rejection of claim 1.

65. Referring to claim 82, the method of claim 82 is performed by the accelerator of claim 68. Consequently, claim 82 is rejected for the same reasons set forth in the rejection of claim 68.

66. Claims 74-79 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nakagoshi in view of Grondalski.

67. Referring to claim 74, Nakagoshi has taught an accelerator comprising:

a) first and second memories. See Fig.9, component 901 and Fig.3, component 203.

b) a hardwired circuit coupled to the first and second memories (see Fig.3, at least components 201) and comprising:

b1) an input-data handler operable to receive from an external source a first message that includes raw data and that includes a first header having information indicating a destination of the raw data, to extract the raw data from the message, and to load the raw data into the first memory. See Fig.5, field 501, and column 11, lines 1-14.

b2) at least one hardwired circuit operable to process data. See Fig.3, components 201.

b3) an interface operable to retrieve the raw data from the first memory. See Fig.9.

b4) provide the retrieved raw data to the hardwired circuit corresponding to the destination. See Fig.5, field 501. This field specifies the destination.

b5) Nakagoshi has not explicitly taught loading processed data from the hardwired pipeline into the second memory. However, the examiner asserts that it is well known that processors, when done processing, store data to memory. This allows a large amount of processed data to be stored, which is useful if needed at some subsequent point during processing. As a result, in order to store results for future use, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi such that the processed data is provided to a second memory (Fig.3, component 203).

b6) Nakagoshi has not explicitly taught an output-data handler operable to retrieve the processed data from the second memory and to provide the processed data to the external source. However, Grondalski has taught an array processor in which after an element performs its processing, it can send the processed data as a message to another element for further processing. See column 3, lines 26-30. Because this is a well-known type of processing, and because it allows for calculation of final results and/or different processing to be implemented in different units, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi in view of Grondalski such that the processed data is retrieved from the second memory and provided to the external source.

b7) Nakagoshi in view of Grondalski has further taught generating a second header having first information indicating a destination of the processed data, and to generate a second message that includes the processed data and the second header. See Fig.5 of Nakagoshi.

c) Nakagoshi has further taught that the input handler is further operable to:

c1) extract from the header the information indicating the destination of the data. See Fig.5.

c2) generate from the extracted information an identifier that identifies the circuit corresponding to the destination. See Fig.5, field 503. This field indicates how to process the data, and consequently, which hardware will be used to process the data.

c3) Nakagoshi has not explicitly taught storing a pointer to the extracted data. However, the examiner asserts that a memory address/pointer is known to be stored somewhere in the system so that memory may be accessed at the location pointed to. That is, it is known at the very least to latch a memory address when using it to access data. As is known, latching allows for synchronous operation which is reliable and deterministic operation. Consequently, in order to achieve synchronous operation, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi such that a pointer to the extracted data is stored.

c4) wherein the interface is further operable to provide the retrieved data to the circuit in response to the stored pointer and identifier. In order for the data to be sent for processing, it must be retrieved from the memory using the stored pointer. And, based on the type of processing, the appropriate hardware is determined. For instance, if integer processing is required, as is well known, the identifier will specify integer processing, and the data will be sent to the integer unit. However, if floating-point processing is required, as is well known, then it will be sent to the floating-point unit.

d) Nakagoshi has not taught a pipelined implementation. However, pipelining is well known and accepted in the art. Pipelining has known advantages such as increasing throughput by overlapping execution of instruction as opposed to serial execution of instructions which is slower. Consequently, in order to increase throughput, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Nakagoshi to be pipelined.

68. Referring to claim 75, claim 75 is rejected for the same reasons set forth in the rejection of claim 74.

69. Referring to claim 76, claim 76 is rejected for the same reasons set forth in the rejection of claim 74.

70. Referring to claim 77, Nakagoshi in view of Grondalski has taught the pipeline accelerator of claim 76. Nakagoshi has further taught that the second information equals the first information. Both the first and second information have been interpreted as being the destination processor information. Hence, they are the same.

71. Referring to claim 78, claim 78 is rejected for the same reasons set forth in the rejection of claim 74, where the processed data must be retrieved from memory using a pointer when sending it on to another array element, and where the second information is field 503 in Fig.5 of Nakagoshi, where this field is used to generate first information as first information is generated in response to processing the data.

72. Referring to claim 79, claim 79 is rejected for the same reasons set forth in the rejection of claim 78.

Response to Arguments

73. Applicant's arguments filed on October 22, 2007, have been fully considered but they are not persuasive.

74. Applicant argues the novelty/rejection of claim 1 on page 22 of the remarks, in substance that:

> "Morikawa does not disclose a hardwired-pipeline circuit operable to receive, via a single bus, a message that includes data. Referring, e.g., to FIG. 1 of Morikawa, a coprocessor 102 receives an instruction via a first bus (i.e., a bus 126), receives data via a second bus (i.e., a bus 124), and processes the received data beginning with a first coprocessing unit 150. Neither the data nor the instruction includes a header. Similar analyses apply to the coprocessors 202 (FIG. 4) and 302 (FIG. 10) of Morikawa. Thus, Morikawa does not disclose or fairly suggest a hardwired- pipeline circuit operable to receive, via a single bus, a message as recited in claim 1 and, instead, teaches that two separate buses are required."

75. These arguments are not found persuasive for the following reasons:

a) As indicated in the rejection of claim 1, the message includes a header because one of the buffer destinations is selected according to information in the message. And, the examiner asserts that Morikawa can be viewed as having a message bus that comprises sub-buses 124 and 126. The message bus, therefore, would be a single bus for transferring a message. Note from the Microsoft Computer Dictionary, 5th Edition (which is cited as extrinsic evidence), that bus is merely defined as "a set of hardware lines used for data transfer among the components of a computer system. A bus is essentially a shared highway that connects different parts of the system...". Consequently, it can be seen that the set of wires 124 and 126 for a bus. As a result, claiming "a single bus" does not overcome Morikawa.

A similar response applies to applicant's argument with respect to claim 6 on page 23 of the remarks.

76. Applicant argues the novelty/rejection of claim 41 on pages 23-24 of the remarks, in substance that:

> "Morikawa does not disclose or fairly suggest the act of receiving a message that includes a header having information indicating a size of the message recited in claim 41."

77. These arguments are not found persuasive for the following reasons:

a) As described in the rejection of claim 41 above, Fig.2 shows that an instruction opcode indicates an immediate value size (imm8, imm16, and imm32) of the message.

78. Applicant argues the novelty/rejection of claim 43 on page 24 of the remarks, in substance that:

> "Morikawa does not disclose or fairly suggest providing a message to an external source via a single bus."

79. These arguments are not found persuasive for the following reasons:

a) As described in the rejection of claim 43 above, the message is provided to an external source via a single bus (Fig.4, bus 125).

80. Applicant argues the novelty/rejection of claim 1 on page 24 of the remarks, in substance that:

> "Hennessy discloses that a register receives instructions via a first bus and data from the data memory via a second bus. The value B appears to be the address of the data in the data memory. Thus, Hennessy does not disclose hardwired-pipeline circuit operable to receive a message via a single bus and, instead requires two separate buses."

81. These arguments are not found persuasive for the following reasons:

a) The value B in Fig.3.15 is not a memory address but an immediate value. This is clear because the instruction sequence is performing A = B + C, where B and C are the values added, not the values at memory addresses B and C. And, even if B and C were memory addresses, a single bus is still just a set of wires that may comprise multiple sub-buses.

82. Applicant argues the novelty/rejection of claim 41 on page 25 of the remarks, in substance that:

> "Hennessy does not disclose receiving a message that includes data and a header having information indicating a size of the message."

83. These arguments are not found persuasive for the following reasons:

a) As discussed in the rejection of claim 41 above, the message has information indicating a size of the message. The "LW" indicates that a <u>word</u> B is to be loaded into R1. Specifically, B is an immediate value that is equivalent in size to a word. The size of B, which is indicated, is a size of the message.

84. Applicant's arguments with respect to claims 7-8 and 43 on pages 25-26 of the remarks have been considered but are moot in view of the new ground(s) of rejection.

85. Applicant argues the novelty/rejection of claim 41 on page 25 of the remarks, in substance that:

> "Morikawa fails to teach or suggest an output-data handler operable to generate a second header having first information indicating a destination of the processed data and to generate a second message that includes the processed data and the second header. Additionally, as previously discussed, there is no reason to modify Morikawa so that the coprocessor 202 generates a message that includes a header having information indicating a destination for processed data."

86. These arguments are not found persuasive for the following reasons:

a) The examiner disagrees with applicant's argument because the examiner has been unable to find any disclosure in Morikawa that the coprocessor does not or cannot send a destination with the destination data. As explained, the data to be sent back to the processor is written to a register. Consequently, a register address must also be specified. This can be done by either the processor or coprocessor, and either is obvious. Applicant states that the processor does it, but the examiner can find no support for such a statement. The processor does do the writing, but the examiner feels it's obvious for the coprocessor to send the data since it executed the instruction and it knows the destination. Such a combination would not break the machine or add complexity. It merely sets forth which circuit provides a register destination address.

A similar response applies to applicant's argument with respect to claim 44 on page 27 of the remarks.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

DJH
David J. Huisman
December 14, 2007

[illegible]